RAPORT DU PROJET : BASE DE DONNEES POUR UN SITE WEB

Ce projet a été réalisé par :

- KALUMVUATI Duramana
- DE LAS HERAS MORENO Martin

Objectif :

Mise en place une base de données pour un site internet permettant aux utilisateurs du site internet de créer des agendas contenant des activités.

Après les utilisateurs pourront partager les activités de leurs agendas. voir aussi celles des autres utilisateurs et de s'inscrire. Et finalement ils pourront évaluer des agendas (à une note comprise entre 1 à 5).

Méthodologie :

Chassant que cette base de données sera utilisée pour un site web, on en deduit que certains traitements des informations seront faits par celui qui va utiliser la base de données mais, on a voulu aussi faire le maximum de traitement possible pour réduire le travail de celui qui s'occupéra de Back-End.

Entités :

Selon les demandes du dévéloppeur Back-End, on a résolu de mettre en place ces entités, capables à répondre aux bésoins cités dans l'énoncé.

Table de rélation entre agenda et ses activités

> ACTIV_AGENDA (**#idActivite**, **#idAgenda**, priorite);

Cette table permettra aux utilisateurs de pouvoir créer plusieurs activites et de fusionner les activités des agendas par exemple à un seul agenda.

Table des activités

> ACTIVITES (**#idActivite**, titre, description,localisation,estPause, #idType, #idUtilisateur);

> ACTIVITESARCHIVEES (**#idActivite**, titre, description,localisation,estPause, #idType,datedebut,datefin,periodicite, #idUtilisateur);

- *activitesarchivees :* Au début, on voulait créer une colonne sur la table activités, pour que chaque fois une activité supprimée, reste sur la même table et juste son état qui change. Nous avons eu quelques problèmes lors de mise en place de trigger qui devait remplir cette tâche. Donc, on a pris la décision, la mise en place de cette table. En effet, ça nous a aidé à faire quelques recherches. Donc, pas de regret, car ça nous a permis d'utiliser d'autre méthode qu'on ne connaissait pas avant.

- *estPause :* Pour cette colonne, toute activité qui se trouve en pause, ne doit pas être prise en compte et ni permettre les inscriptions sur cette dernière.

Table des agendas

> AGENDAS (**#idAgenda**, nomAgenda, estArchive, dateCreation, dateModification, #idUtilisateur,allowsimult);

> AGENDASARCHIVES (**#idAgenda**, nomAgenda, estArchive, dateCreation, dateModification, #idUtilisateur,allowsimult);

Cette table permet de créer des agendas contenant le nom, la date de création, la date de la dernière modification, et l'id de l'utilisateur qui l'a créé.

- *allowsimult :* Colonne très importante lors qu'il faudra interdire la simultanéité de date de certaines activités.

Table des evaluations

> EVALUATIONS (**#idAgenda**, **#idUtilisateur**, note,dateevaluation);

Cette table permet aux utilisateurs de noter les agendas, et on souhait aussi garder la date d'évaluation. Pour pouvoir interdir à certain utilisateur d'évaluer un agenda. Pour le respect du besoin.

Table de rélation entre utilisateur et activité

> INSCRIPTION_UTIL_ACT(**#idUtilisateur**, **#idActivite**);

Cette table permet de voir quel utilisateur est inscrit dans quelle activité (si un utilisateur est inscrit dans un agenda).

Table de rélation entre utilisateurs et agendas

> INSCRIPTION_UTIL_AGENDA (**#idAgenda**, **#idUtilisateur**, estAbonne);

Cette table permet de voir les utilisateurs qui sont inscrits dans chaque agenda, et aussi avec *estAbonne* on peut savoir si l'utilisateur est abonné dans cet agenda, par défaut est mis à 0 (pas inscrit).

Table des occurences

> OCCURENCES (**#idOccurence**, #idActivite, dateoccurence);

Elle permet de voir toutes les occurences des activités (car les activités peuvent avoir une periodicité et apparaître plusieurs fois). Elle est remplie automatiquement lors d'insertion d'une activité, enfin de garder la cohérence des activités et leurs périodicités.

Table des types

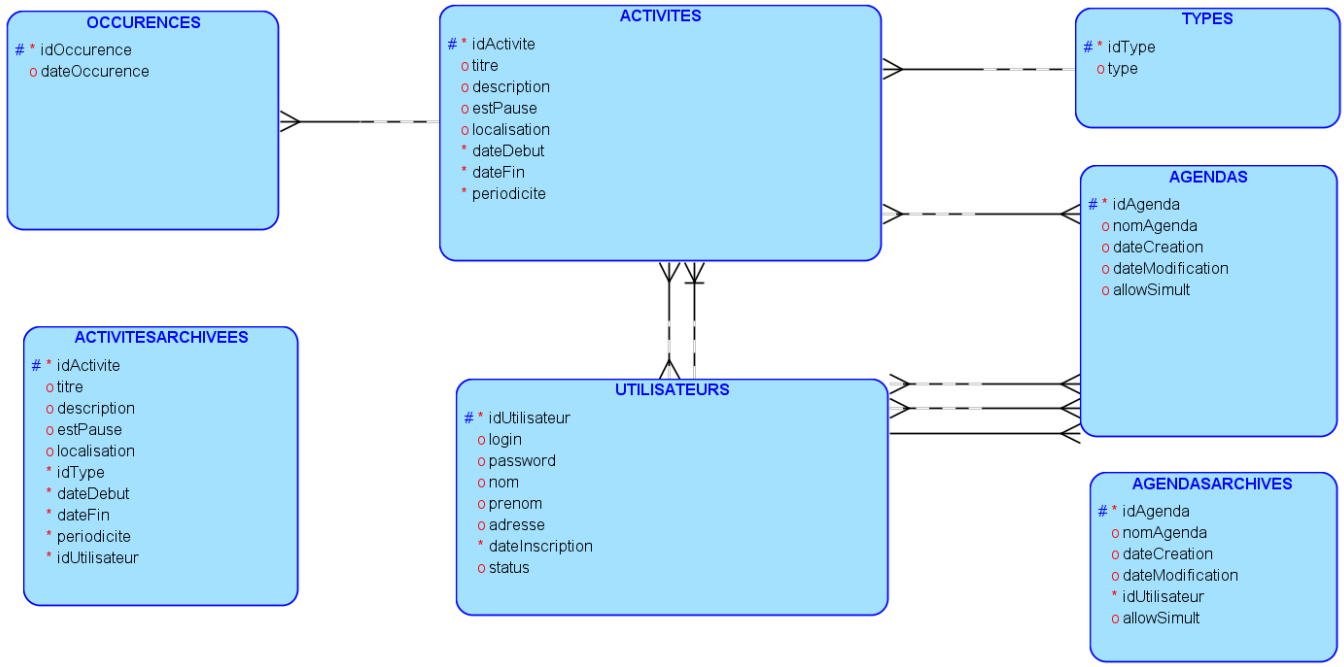
> TYPES (**#idType**, type);

Table des catégories de activités.

Table des utilisateurs

> UTILISATEURS (**#idUtilisateur**, login, motdepass, nom, prenom, adresse,dateInscription,status);

Cette table permet de voir tous les utilisateurs. Et la colonne "status" nous permet de savoir si l'utilisateur est juste un utilisateur simple ou administrateur, enfin de lui permettre à modifier certaines activités ou agendas.



Contraintes:

Cette contrainte vérifie sur la table *evaluations* que la note donnée par l'utilisateur est entre 1 et 5.

```
ALTER TABLE evaluations ADD
    CONSTRAINT evaluations_ck CHECK (note between 1 and 5);
```

Cette contrainte vérifie l'unicité d'un login.

```
ALTER TABLE utilisateurs ADD CONSTRAINT utilisateurs_u UNIQUE ( login );
```

Cette contrainte référencie la colonne *idType* dans la table *activites*.

```
ALTER TABLE activites
    ADD CONSTRAINT activites_types_fk FOREIGN KEY ( idtype )
        REFERENCES types ( idtype ) ON DELETE CASCADE;

ALTER TABLE activites ADD CONSTRAINT activites_titre_u UNIQUE ( titre );

ALTER TABLE activites
    ADD CONSTRAINT activites_util_fk FOREIGN KEY ( idUtilisateur )
        REFERENCES utilisateurs ( idUtilisateur ) ON DELETE CASCADE;
```

Elle permet qu'une activité appartienne à plusieurs agendas, et un agendas aie plusieurs activités

```
ALTER TABLE activite_agenda
    ADD CONSTRAINT activite_agenda_act_fk FOREIGN KEY ( idactivite )
        REFERENCES activites ( idactivite ) ON DELETE CASCADE;

ALTER TABLE activite_agenda
    ADD CONSTRAINT activite_agenda_ag_fk FOREIGN KEY ( idagenda )
        REFERENCES agendas ( idagenda ) ON DELETE CASCADE;
```

Cette contrainte référencie la colonne *idUtilisateur* de la table *agendas* à la même dans la table *utilisateurs*.

```
ALTER TABLE agendas ADD CONSTRAINT agendas_nomagenda_u UNIQUE ( nomagenda
);
ALTER TABLE agendas ADD CONSTRAINT agendas_ck CHECK (allowSimult BETWEEN 0
AND 1);

ALTER TABLE agendas
    ADD CONSTRAINT agendas_util_fk FOREIGN KEY ( idutilisateur )
        REFERENCES utilisateurs ( idutilisateur ) ON DELETE CASCADE;
```

Un utilisateur peut évaluer un ou plusieurs agendas.

```
ALTER TABLE evaluations
    ADD CONSTRAINT evaluations_agendas_fk FOREIGN KEY ( idagenda )
        REFERENCES agendas ( idagenda ) ON DELETE CASCADE;

ALTER TABLE evaluations
    ADD CONSTRAINT evaluations_utilisateurs_fk FOREIGN KEY ( idutilisateur
)
        REFERENCES utilisateurs ( idutilisateur ) ON DELETE CASCADE;
```

Cette contrainte réfferencie une activité dans la table occurences

```
ALTER TABLE occurences
    ADD CONSTRAINT occurences_activites_fk FOREIGN KEY ( idactivite )
        REFERENCES activites ( idactivite ) ON DELETE CASCADE;
```

Un utilisateur peut s'inscrire à plusieurs activités

```
ALTER TABLE inscrip_util_act
    ADD CONSTRAINT inscrip_util_act_act_fk FOREIGN KEY ( idactivite )
        REFERENCES activites ( idactivite ) ON DELETE CASCADE;

ALTER TABLE inscrip_util_act
```

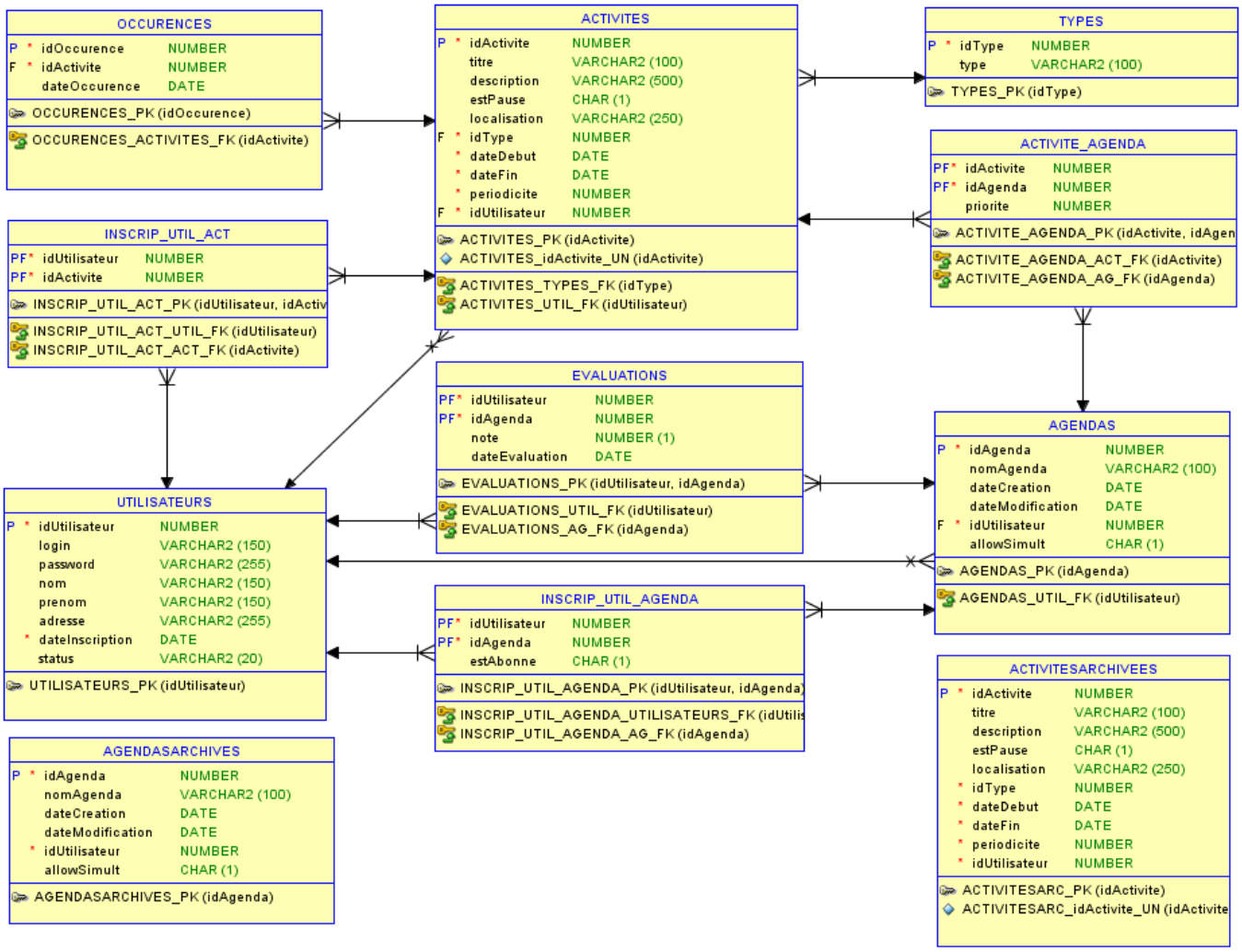
```
    ADD CONSTRAINT inscrip_util_act_util_fk FOREIGN KEY ( idutilisateur )
        REFERENCES utilisateurs ( idutilisateur ) ON DELETE CASCADE;
```

Elles permettent à un utilisateur de s'inscrire à plusieurs agendas

```
ALTER TABLE inscrip_util_agenda
    ADD CONSTRAINT inscrip_util_agenda_ag_fk FOREIGN KEY ( idagenda )
        REFERENCES agendas ( idagenda ) ON DELETE CASCADE;

ALTER TABLE inscrip_util_agenda
    ADD CONSTRAINT inscrip_util_agenda_util_fk FOREIGN KEY ( idutilisateur
)
        REFERENCES utilisateurs ( idutilisateur ) ON DELETE CASCADE;
```

```
-- Mise en place la séquence, pour tester l'occurence
```

```
CREATE SEQUENCE SeqOccurence
START WITH 1
INCREMENT BY 1
CACHE 100 ;

SET SERVEROUTPUT ON

/*
    -- Trigger qui assure la cohérence entre les activités et le nombre
d'occurences
*/
CREATE OR REPLACE TRIGGER testActiviteOccurence
AFTER INSERT ON activites
FOR EACH ROW
DECLARE
compteur_v INTEGER := 1;
BEGIN
LOOP
    INSERT INTO occurences VALUES(SeqOccurence.nextval, :new.idactivite,
sysdate);
    EXIT WHEN compteur_v >= :new.periodicite;
    compteur_v := compteur_v + 1;
END LOOP;
DBMS_OUTPUT.put_line('INSERTION DES ACTIVITES AVEC SUCCES, VEILLEZ MODIFIER
LES DATES DES OCCURENCES');
END;
/
```


OCCURENCES
P * idOccurence NUMBER
F * idActivite NUMBER
dateOccurence DATE
OCCURENCES_PK (idOccurence)
OCCURENCES_ACTIVITES_FK (idActivite)
ACTIVITES
P * idActivite NUMBER
titre VARCHAR2 (100)
description VARCHAR2 (500)
estPause CHAR (1)
localisation VARCHAR2 (250)
F * idType NUMBER
* dateDebut DATE
* dateFin DATE
* periodicite NUMBER
F * idUtilisateur NUMBER
ACTIVITES_PK (idActivite)
ACTIVITES_idActivite_UN (idActivite)
ACTIVITES_TYPES_FK (idType)
ACTIVITES_UTIL_FK (idUtilisateur)
TYPES
P * idType NUMBER
type VARCHAR2 (100)
TYPES_PK (idType)
ACTIVITE_AGENDA
PF* idActivite NUMBER
PF* idAgenda NUMBER
priorite NUMBER
ACTIVITE_AGENDA_PK (idActivite, idAgen
ACTIVITE_AGENDA_ACT_FK (idActivite)
ACTIVITE_AGENDA_AG_FK (idAgenda)
INSCRIP_UTIL_ACT
PF* idUtilisateur NUMBER
PF* idActivite NUMBER
INSCRIP_UTIL_ACT_PK (idUtilisateur, idActiv
INSCRIP_UTIL_ACT_UTIL_FK (idUtilisateur)
INSCRIP_UTIL_ACT_ACT_FK (idActivite)
EVALUATIONS
PF* idUtilisateur NUMBER
PF* idAgenda NUMBER
note NUMBER (1)
dateEvaluation DATE
EVALUATIONS_PK (idUtilisateur, idAgenda)
EVALUATIONS_UTIL_FK (idUtilisateur)
EVALUATIONS_AG_FK (idAgenda)
AGENDAS
P * idAgenda NUMBER
nomAgenda VARCHAR2 (100)
dateCreation DATE
dateModification DATE
F * idUtilisateur NUMBER
allowSimult CHAR (1)
AGENDAS_PK (idAgenda)
AGENDAS_UTIL_FK (idUtilisateur)
UTILISATEURS
P * idUtilisateur NUMBER
login VARCHAR2 (150)
password VARCHAR2 (255)
nom VARCHAR2 (150)
prenom VARCHAR2 (150)
adresse VARCHAR2 (255)
* dateInscription DATE
status VARCHAR2 (20)
UTILISATEURS_PK (idUtilisateur)
INSCRIP_UTIL_AGENDA
PF* idUtilisateur NUMBER
PF* idAgenda NUMBER
estAbonne CHAR (1)
INSCRIP_UTIL_AGENDA_PK (idUtilisateur, idAgenda)
INSCRIP_UTIL_AGENDA_UTILISATEURS_FK (idUtilis
INSCRIP_UTIL_AGENDA_AG_FK (idAgenda)
ACTIVITESARCHIVEES
P * idActivite NUMBER
titre VARCHAR2 (100)
description VARCHAR2 (500)
estPause CHAR (1)
localisation VARCHAR2 (250)
* idType NUMBER
* dateDebut DATE
* dateFin DATE
* periodicite NUMBER
* idUtilisateur NUMBER
ACTIVITESARC_PK (idActivite)
ACTIVITESARC_idActivite_UN (idActivite
AGENDASARCHIVES
P * idAgenda NUMBER
nomAgenda VARCHAR2 (100)
dateCreation DATE
dateModification DATE
* idUtilisateur NUMBER
allowSimult CHAR (1)
AGENDASARCHIVES_PK (idAgenda)